REDUCTION OF LEXICAL AMBIGUITY

Éric Laporte
IGM, University of Marne-la-Vallée - CNRS

The resolution of lexical ambiguity is a prerequisite for many automatic procedures on written texts, even simpler ones. However, it is not an easily automatable task. We will examine on concrete examples the issues faced during the elaboration of lexical disambiguators. In order to estimate the potential of approaches, we will consider how disambiguating written texts before processing them brings about improvements to relevant applications. In this study we will take into account both linguistic and computational problems and show how they are connected[1].

1. Lexical tagging of texts

Written text cannot undergo linguistic processing without the system having access to linguistic information about words. In order to make such information quickly and conveniently available, computer programs usually attach it to the words of the text themselves in the form of lexical tags. The lexical tag for a word, therefore, gathers all the information available about it and useful for the task to be performed, ranging from the very form occurring in the text to grammatical, morphological, syntactic and semantic data, according to the nature of the task. A basic step consists in segmenting the text, identifying minimal units and annotating them with tags. This task is called lexical analysis, lexical tagging or annotation.

The technical means of attaching lexical information to words can be classified into two types, depending on whether the information comes from an electronic dictionary or is deduced from information present in the text.

Dictionary-based tagging is simple: the program looks up the words in a dictionary that associates tags to all the words in the language. This approach was widely put to the test in the 1990's and yields the most reliable results, in so far as the dictionary conforms to actual usage of the language and is comprehensive enough. For inflected languages, like most European languages, inflected dictionaries are used. The number of entries in inflected-form dictionaries is larger than in conventional dictionaries, in which verbs are present only in the infinitive. Highly inflected languages, e.g. Polish, have several millions of inflected words. Even so, there exist dictionaries reasonably close to exhaustivity, that can be compressed into files of an order of magnitude of 1 Mb, making it possible to tag thousands of words per second. The Intex system contains efficient tools for dictionary compression and text tagging (M. Silberztein 1994). In this article, we will use usual Intex conventions for lexical tags: thus, in French, <*actif,A:fp*> represents the adjective *actif* in the feminine plural, i.e. *actives*.

Approaches to tagging without dictionary were implemented in numerous systems during the 1980's and 1990's. Such systems exploit information present in the text, such as final parts of words and contexts. For example, many French words in *-ives* are adjectives in the feminine plural. This rule correctly assigns the tag <*A:fp*> to the word *actives* in the sentence:

Les entreprises les plus actives ont gagné des parts de marché

[1] This work was partially financed by the European Union project Copernicus 621 Gramlex. A former version of this study is published in Portuguese as a chapter in E. Ranchhod (ed.), 2001, *Tratamento das Línguas por Computador. Uma introdução à Linguística Computacional e suas Aplicações*, Lisbon: Caminho.

Many words preceded by *certains des* are nouns in the masculine plural. This rule, when applied to the sentence:

Certains des films primés ne seront à l'affiche qu'en automne

correctly assigns the tag <*N:mp*> to *films*. Such rules are automatically learned through a training based on frequency counts in a sample of tagged or untagged texts. Several types of machine learning are known. The principle itself is an approximation, since words with the same final part may have entirely different properties, for example the adjective *moindre* and the verb *joindre*. However, it is the only solution for tagging unknown words automatically.

2. Lexical ambiguity

Residual difficulties with lexical tagging result from the massive density of lexical ambiguity in all natural languages. There is an instance of lexical ambiguity between two distinct linguistic elements when they are spelled exactly the same way. Some instances of lexical ambiguity come from imperfections in writing systems of languages. For example, in the two sentences:

Les opinions divergent sur ce point
On place sur le trajet du faisceau lumineux un dispositif divergent

the two forms *divergent*, a verb and a noun, which are pronounced differently, are identical only because of oddities of French spelling. This type of ambiguity is more or less abundant, according to writing systems. Theoretically, it would not exist in the case of a language with a sufficiently informative writing system.

However, lexical ambiguity covers a much more extensive reality than mere curiosities and sporadic contingencies. We will examine several examples of ambiguous words. In the following sentences, the two occurrences of *forme* are a noun and a verb, although they do not present any phonetic difference:

On ne forme pas un spécialiste en quelques semaines
Le contrat prend une forme de nature à satisfaire les deux parties

A human reader does not perceive the slightest ambiguity, but the necessity of assigning different tags to a verb and a noun, in a step which comes before recognition of syntactic structures of sentences, for example, is obvious. Two terms of an instance of lexical ambiguity may have an etymological relation, like *forme*, or not, like *peignait*:

(1) *Luc peignait les cheveux d'Anne*
Luc peignait ses volets en vert

They can even be two elements of the same paradigm:

Je sais que vous plaisantiez
J'aime que vous plaisantiez

In this example, the two occurrences of *plaisantiez* differ in inflectional features: respectively indicative imperfect and subjunctive present. Lexical ambiguity often involves two words with a close etymological relation, the same part of speech, but two senses:

(2) *Luc a marqué une croix sur cette page*
(3) *Ce livre m'a profondément marqué*

Syntactic properties depend on senses: in the case of verbs, the number of essential complements, the corresponding prepositions, the syntactic transformations applicable, the selectional restrictions on subjects and essential complements... may differ. Now, such syntactic information, which is required for recognizing sentence structures, must be included in lexical tags for several classes of applications. We will come back to this topic in section 3.

Another common type of lexical ambiguity involves compound words (M. Salkoff 1999):

(4) *La table ronde sur la politique de santé est annulée*
(5) *La table ronde est trop petite, il faudra prendre l'autre*

In (4), *table ronde* is a compound noun and, as such, a minimal unit for linguistic processing, therefore the tag for *table ronde* will have to contain relevant syntactic information, e.g. the form and type of complements of this unit. In (5), *table* and *ronde* are distinct units that make up a free noun phrase. The tradition in natural language processing is persistently reluctant to admit the idea of representing a compound by a single tag (e.g. J. Cloeren 1999, p. 45). The computational linguistics community quite as persistently underestimates the quantity of compound words in texts, though recent studies (e.g. J. Senellart 1999) estimate at more than a half the proportion of texts constituted by compound words. In addition, the technical content of a text, including most technical terms, is to be found more in this part than in the other. Lexical tagging should deal with units of greater complexity than simple words. Tagging only simple words amounts to restricting all further computer processing to a superficial part of language.

Natural languages also provide examples of other types of ambiguity besides lexical ambiguity (e.g. M. Gross, in this volume), but since we are not interested in them in this article, we will most of the time just refer to ambiguity for short.

The existence of lexical ambiguity has technical consequences. The function of an electronic dictionary is to ensure that relevant tags are available for all words; in the case of multiple senses, therefore, a mere dictionary lookup yields lexical information about all senses, and not only the contextually appropriate information. At this stage, ambiguity is explicitly represented through tag lists; it must be resolved in order for the lexical tagging process to be complete, i.e. for each word to be annotated only with the right tag — or tags. Indeed, several tags may perfectly be contextually appropriate. It is the case of sentence (1), in which Luc can have either made a painting of Anne's hair or combed it. It is an example where ambiguity can be represented at the level of lexical tagging but cannot be resolved (genuine ambiguity).

The case of tagging without dictionary is different: it is possible to produce several tags in order to represent several analyses, but it is impossible to ensure that the correct tagging of each word is present among the tags produced. In addition, recognizing compound words without a dictionary is an almost insoluble problem, since most of them obey all usual rules of morphological agreement and superficial syntax, like *table ronde, prendre en compte, au cours de,* and therefore are impossible to detect without complete lists. To our view, this limitation completely disqualifies the exclusive use of statistical taggers for useful tagging. From now on we will restrict our scope to dictionary-based tagging.

Electronic dictionaries may be more or less comprehensive, in the sense that they describe a larger or smaller number of words, or of word senses. When the number of senses

described increases, the degree of ambiguity also grows, which can appear as an obstacle to computer processing in the case of rare senses, like *oublie* 'sort of cake', an old-fashioned word and a homograph of forms of *oublier* 'forget'. For some applications, this is an argument for limiting the extension of the lexical description, but the fundamental work of describing linguistic data cannot be restricted by considerations specific to a particular type of application: the description of rare words and rare senses is required for other applications. Restricting lexical description in order to prevent ambiguity from becoming visible would be equivalent to stopping the development of air traffic in order to avoid building airports.

3. Granularity

Lexical information found in electronic dictionaries and attached to words may be more or less detailed, according to its nature and extension. For instance, tags reduced to an indication of grammatical category: verb, noun, adjective... are little informative. The following sentence is an example of such a minimalist tagging, with classical abbreviations:

(6) *Il<PRO> pense<V> arriver<V> à l'heure<ADV>*

The informative content of tags is related with the size of the set of tags: parts of speech make up a set of about 15 elements. If we take into account more information, this amount can only grow. This quantitative parameter is called the granularity of the description, since in proportion as it grows, each tag describes less words in a more informative way, and the descriptive model gets more fine-grained.

The granularity of a description grows when more information is included. The most popular tagsets for English include essentially, in addition to part of speech, inflectional features: tense, number, person... The British National Corpus basic tagset totals 57 lexical tags (G. Leech et al. 1994), the Penn Treebank tagset, 36 (M. Marcus et al. 1993), the Lancaster-Oslo-Bergen Corpus tagset, 120 (S. Johansson et al. 1986; Garside et al. 1987), and the Brown Corpus tagset, 71 (B. Greene & G. Rubin 1971, N. Francis & H. Kucera 1982). In Romance languages, which are more inflected than English, the variations of the same features generate 80 to 120 tags[2]. Let us insert inflectional features into sentence (6):

(7) *Il<PRO:3ms> pense<V:P3s> arriver<V:W> à l'heure<ADV>*

We can still include into the tags the canonical form (or lemma or base form) of each inflected word, e.g. the infinitive in the case of verbs:

(8) *Il<il,PRO:3ms> pense<penser,V:P3s> arriver<arriver,V:W> à l'heure<à l'heure,ADV>*

When canonical forms are included in all tags, the number of tags increases dramatically: the class of forms to which a given tag can be assigned usually reduces to a single word. Common tagsets do not include canonical forms of words beyond a few grammatical words, and are often called *wordclass tagsets*. It is natural to avoid this term in a framework where wordclasses are very small. Since we consider that canonical forms are an important piece information, we will prefer the term of *lexical tagsets*.

To limit the number of tags without any loss of information, canonical forms can be abbreviated, taking advantage of the redundancy between inflected form and canonical form:

[2] In fact, common tagsets for English include canonical forms of a few grammatical words. If the same convention were adopted in the case of Romance languages, our number of tags should be roughly doubled.

with this convention, *pensait<penser,V:I3s>* is replaced with *pensait<3er,V:I3s>*. The amount of distinct tags reaches an order of magnitude of 1000 in French. The morphological dictionaries of the RELEX network of laboratories, which are integrated in Intex, use tagsets of at least this level of granularity.

In examples (6)-(8), we included into tags grammatical, morphological and inflectional information only. Even so, the granularity of lexical description varied greatly. The representation of ambiguity depends on the system of tags. The word *pense* was not represented as ambiguous in (6):

pense<V>

The same word becomes ambiguous as soon as we take into account inflectional features, since it can be the first or third person in the indicative present, or the same in the subjunctive, or the second person in the imperative:

pense<V:P1s>
pense<V:P3s>
pense<V:S1s>
pense<V:S3s>
pense<V:Y2s>

Thus, ambiguity grows automatically with the granularity of the tagset used to represent it. However, statistics on texts show that this growth is moderate as long as information is restricted to the grammatical, morphological and inflectional level. We computed that the average lexical ambiguity of a sample of French texts is 1.63 tags per word with a tagset of the type of (6), and 1.99 with one of the type of (8). The growth of ambiguity is limited (22%) as compared with the increase in informative content brought about by the substitution of tagsets.

From the point of view of application, the adequacy of tagsets for the particular application must first be assessed. The simplest procedure on written text is the detection of lexical errors, i.e. of words that do not belong in the vocabulary, like *pourparler* (for *pourparlers*): this task does not require any tags. Some more ambitious applications require tagging the text, but can be considered acceptable even if they do not produce exhaustive results. For example, users of spelling checkers for non-lexical errors are usually aware of the difficulty of the task, view such systems as an aid to re-reading text and do not trust them to exhaustively point errors. The same holds for text searching and text indexing. The texts selected, in the first case, and the index items chosen, in the second case, can include undesired elements and miss desired elements, without making such systems unusable, since the user's purpose is only approximately defined and output is processed manually. For this category of applications, the information in tags of the type of (8) probably allows for obtaining more interesting results, as compared to the present state of the art.

More ambitious applications require even more lexical information: speech synthesis from unrestricted written text, translation, and other applications involve thorough syntactic parsing, i.e. recognition of the structures of sentences and phrases: clauses, predicates, complements. This recognition cannot be automated without specific lexical information. Thus, surveying the applications which tagging is useful for leads to examine the relations between tagging and syntactic parsing. It is a common prejudice to consider a priori that these two procedures belong to distinct, independent areas. Let us instead examine which kind of information required by syntactic parsing must be included in lexical tags.

In the case of verbs and other predicative forms, like *marquer, faire un calcul, être d'accord, être débiteur, prendre en compte*, the number of essential complements and the corresponding prepositions are indispensable for identifying the predicative element(s) of the sentence and their arguments: subjects and complements, if any. In addition, the sentence can be the result of the application of one or several syntactic operations: passive, inversions, reductions, omissions, pronominalizations... Now, specialists of syntax know that not all transformations are applicable to all predicative forms, and this information is essential for recognizing structures. Selectional restrictions, i.e. the information of the set of nouns that can occur in a sentence as subjects or complements, depend on each predicative form: it is fundamental to know them, even incompletely, to discriminate hypotheses during computer parsing, e.g. in (2) or (3).

Formalization of such properties is an extensive work, since lexicon is concerned. The Lexicon-Grammar theory (M. Gross 1994a, 1994b) is the natural framework for implementing such a study.

Let us examine a simple example of a linguistic piece of information required for syntactic parsing: the pre-nominal and post-nominal position of adjectives in French. This information is lexical, in the sense that it depends on each adjective and does not comply with any general rules. The adjective *sympathique* may be pre- or post-nominal, the adjective *politique* is post-nominal:

une sympathique proposition
une proposition sympathique
* *une politique décision*
une décision politique

Both words *adulte* and *analphabète* are ambiguous between an adjective and a noun:

Nous proposons un cours pour adultes analphabètes

In this sentence, the combination of the words *adultes analphabètes* with two parts of speech each generates four analyses; the fact that the adjective *adulte* is exclusively post-nominal correctly rules out the analysis *adultes<A:mp> analphabètes<N:mp>*. The effect of this type of rule on ambiguity resolution in French (M. Garrigues 1997) and in Portuguese (P. Carvalho, this volume) is quantitatively important.

Tags with the kind of linguistic information exemplified above could legitimately be called 'syntactic wordclass tags' or 'syntactic lexical tags'. Such information is indeed of a syntactic level, and is far more precise than the 'morphosyntactic' or 'wordclass' information included in common tagsets (e.g. N. Francis & H. Kucera 1982, R. Garside et al. 1987, M. Marcus et al. 1993, G. Leech et al. 1994). Such tagsets do include more information than bare part of speech; e.g. sometimes they mark a distinction between common nouns and proper nouns, or between auxiliaries and non-auxiliary verbs; but they do not include the classification of verbs into intransitive, direct transitive and indirect transitive — still less the argument structure of verbs. Even intermediate tagsets, designed to ensure comparability between tagsets, do not anticipate the possibility of considering this classification or this structure (G. Leech & A. Wilson 1999). Moreover, as we already mentioned, common tagsets do not systematically include information about canonical forms.

When information exploitable in syntactic parsing is included, the informative content and granularity of tags grow dramatically, since making a formal description of these properties implies separating the senses of verbs, like in (2) and (3), adjectives, and other predicative elements. At this stage, the granularity of the tagset goes far beyond present

standards. The properties that we mentioned are difficult to handle in frequency-based systems. For instance, the pre- or post-nominal position of French adjectives has no correlation with orthographic marks like suffixes, it can depend on the senses of a given form, and often varies freely for a given sense, which makes it difficult to obtain this information through automatic generalization from examples in a sample of texts.

A consequence of an increase in granularity of the description is an increase in ambiguity: each word with several senses associated with several syntactic behaviours can a priori be represented by respective tags, and this kind of ambiguity combines with inflectional ambiguity that we exemplified first. There are no estimates of the average number of tags per word yet, but an order of magnitude of 10 is plausible. However, increase in ambiguity is not a defect of the model, but a reflection and a consequence of the complexity of the problem. A fine-grained tagset is a heavy tool from a technical point of view, but for the same reason why bulldozers are heavy: teaspoons are not appropriate tools for making embankments.

As far as degree of lexical ambiguity is concerned in general, the only technical means of measuring this quantity is to count the amount of tags per word, and this value depends on the granularity of the tagset. Therefore, comparing error rates or reduction rates — i.e. the proportion of ambiguity resolved by respective programs — across systems, or other numerical results of systems with different tagsets, is in general meaningless.

4. Delimitation of the problem and objectives

We examined several examples of lexical ambiguity and we observed that the phenomenon affects any text; and almost any sentence, even the simplest. In addition, depending on the granularity of descriptive models, the degree of ambiguity and the number of tags may increase outstandingly, and must increase due to the complexity of the most interesting applications that justify the study of the problem.

Now, can all lexical ambiguity in a text be resolved, and at which cost?

The answer is clear. For some sentences, resolving all lexical ambiguity would imply recognizing the entire syntactic structure. In the following sentences, *pêcher* is respectively a verb and a noun:

(9) *Il avait l'impression de pêcher dans un aquarium*
(10) *La photo montre un exemple de pêcher dans un terrain siliceux*

Immediate context is analogous in the two sentences, on the left: *<DET:s> <N:s> <de,PREP>* as well as on the right: *<dans,PREP> <un,DET:s> <N:s>*. We need to know the properties of the nouns *impression* and *exemple* in order to resolve the ambiguity. The complement of *impression* is an infinitival clause:

(11) *Luc a l'impression de rêver*

The nominal form of this complement:

Luc a l'impression d'un rêve

may occur in the singular with an empty determiner:

Luc a l'impression de rêve dont il a déjà parlé

but if the complement is a predicative noun, not a concrete noun:

(12) * *Luc a l'impression d'arbre dont il a déjà parlé*

The noun *exemple* does not accept an infinitival complement:

(13) * *Ce texte est un exemple de décrire nos résultats*

The nominal complement of *exemple* behaves according to different restrictions on the use of determiners:

? *Ce texte est un exemple d'une description scientifique*
Ce texte est un exemple de description scientifique
(14) *Ce document est un exemple de papyrus réemployé*

Syntactic constraints (12) and (13), if formalized and available in the dictionary, rule out respectively <*N:ms*> from (9) and <*V:W*> from (10); on the other hand, constructions (11) and (14) respectively justify the choice of <*V:W*> in (9) and of <*N:ms*> in (10).

The pre- and post-nominal positions of adjectives are other cases where the total resolution of lexical ambiguity requires a much more detailed syntactic parsing than usual. There exist sentences for which the same task requires a thorough recognition of the whole syntactic structure.

Thus, a valid determination of the tags to be attached to words may depend on the recognition of global sentence structure. This is a circular dependency, since the recognition of syntactic structures is based on lexical information included in tags. This circular dependency is precisely one of the intrinsic difficulties of computer parsing.

The observation that exhaustive resolution of lexical ambiguity depends, in general, on global syntactic parsing, radically affects the nature of the problem. Correct tagging is a by-product of parsing. Thus, ambiguity resolution does not appear to have an object and a solution as a distinct problem: it disappears as a specific problem.

However, an objective of partially resolving, or reducing, lexical ambiguity, when a thorough syntactic parsing is not required, is less ambitious and more realistic. This goal defines a distinct task: filtering the output of dictionary-based tagging, and removing invalid analyses before parsing or application-specific procedures, or before the intervention of linguists building a syntactic parser[3]. Before filtering, this output is a set of analyses or readings of the text or sentence, and each of these analyses is represented as a sequence of tags generated by dictionary lookup. This process of filtering or selection is meant to facilitate the parsing of the text, by limiting the number of alternative readings of a sentence and the complexity of the data transmitted to the parser, which will produce identical output more efficiently (R. Milne 1986). Taggers and syntactic parsers that work by discarding analyses are said to be reductionistic (A. Voutilainen & P. Tapanainen 1993).

Measures of performance of ambiguity reduction systems ought to be consistent with this goal. They should depend on the reduction of the number of alternative analyses, or on the complexity of the information handed on to the syntactic parser, in order to measure the interest of the method.

If we want to measure the interest of ambiguity reduction independently of the particular parser, or of whatever system to be run after disambiguation, the reduction of the number of alternative readings is obviously the most natural quantity to be used, provided that valid readings are not discarded in the process of filtering.

[3] We consider that a syntactic parser is a set of programs but also of linguistic data, including a syntactic grammar.

However, if we want to take into account more accurately the application-related context of ambiguity reduction, we have to measure how much this procedure speeds up further computer processing. Then, in the case of parsing, the method of measuring depends on the algorithmic content of the parser, and namely of the relation between its input and its execution time.

If the time spent in parsing is in proportion to the number of analyses in input, quickly filtering input is likely to speed up the operation, since the number of analyses of a sentence grows exponentially with the average number of tags per word, and therefore grows very quickly with lexical ambiguity.

The execution time of modern parsing algorithms[4] depends in a complex way, not only on the number of analyses in input, but also on the complexity of the data structure that represents these analyses. This structure can be a finite automaton. Now, when a finite automaton represents a set of sequences, several measures of the complexity of the automaton are known, but none of them is equivalent to the number of sequences. When there is only one sequence, the automaton is obviously bound to be small; but this result is not always within reach; and when some of the sequences are removed during a process of filtering, the complexity of the automaton may increase or decrease. This is why the complexity of the automaton cannot be used as a quantitative means of measuring the performance of the process. And ambiguity reduction can theoretically either speed up or slow down computer parsing.

According to our experimentations in French (É. Laporte & A. Monceaux 1999), ambiguity reduction is quick and generally brings about a dramatic decrease in the complexity of the automaton that represents the alternative analyses of a sentence. It is plausible, therefore, that applying good disambiguators should speed up parsing. However, this hypothesis will have to be empirically checked when satisfactory parsers are available and able to exploit the content of reasonably informative tags. Theoretically, inserting a filtering phase could indeed slow down the global process, and such an operation could become completely useless in the long run.

But such is not the case yet. On the contrary, it is to be expected that the availability of good disambiguators will facilitate the development of parsers.

Therefore, we will define ambiguity reduction as a procedure applied to analyses resulting from tagging, and aiming at rejecting the largest possible number of wrong analyses with the simplest and quickest possible means.

This definition has several important consequences.

First, ambiguity reduction does not make sense but in combination with another task, such as syntactic parsing, and in this case, one observes a strict coupling between ambiguity reduction and parsing, in the sense that one cannot automate the former procedure disregarding how the latter is automated, and that two computer systems that perform the respective procedures in a compatible way cannot be modified independently. For example, the two systems must use the same set of tags and be based on the same type of analyses. Since the result of ambiguity reduction is handed on to the parser, the reliability of the latter depends on the reliability of the former. We will come back to this issue in section 7.

Second, the definition of objectives is vague, in the sense that they include a limitation of the means of achieving the task. This characterizes the problem as application-related, as opposed to more fundamental problems as lexical description or computer parsing, which have definite objectives. Thus, measuring the performance of a disambiguator is not a theoretical, but a purely empirical enterprise. The little theoretical aspect of the problem may reduce the motivation for studying it, but connections with applications are a compensation. In addition, the vagueness of objectives does not imply at all that the task is easy or that a

[4] This situation might in all probability be maintained with future algorithms.

solution can be the result of a rough-and-ready work. The paradox is only apparent: for successful interfacing with such a complex system as a parser, a disambiguator must produce output of an excellent quality.

Third, in consequence of the division of computer processing between an ambiguity reduction step and a syntactic parsing step, what is not achieved during the former must be achieved during the latter. For example, it falls to the parser to resolve all ambiguity remaining in the end of the first step. The motivation for this division is to limit the first step to simple and quick means. For the global organization of the system, this limit must first be set in a more detailed way. The general framework mentioned above implies that the ambiguity reduction step:

- does not involve the systematic recognition of constituents,
- does not involve the insertion or use of any boundary symbols, except sentence boundaries,
- does not generate new readings in addition to those directly produced by dictionary-based tagging,
- does not explicitly represent in output the syntactic transformations that have been applied,

though parsing is certain to resort, in some way or another, to each of these technical means. These limitations are imposed to ambiguity reduction in order to keep it simple and quick. They can also be stated by asserting that this procedure is a filtering and that the context analysed can only be local. For instance, consider the ambiguity of *tours*:

(15) *Il y a deux types de tours médiévales, d'après mon expérience: les défensives et les décoratives*

Several predicative senses ('going round', 'trip', 'turn', 'ballot'…) and a technical sense ('lathe') correspond to the masculine: <*tour,N:mp*>, and the architectural sense ('tower') to the feminine: <*tour,N:fp*>. A local agreement rule between adjacent noun and adjective can make use of the fact that *médiévales* is unambiguously in the feminine in order to correctly choose the tag <*tour,N:fp*>. If the adjective *médiévales* did not occur in the sentence:

Il y a deux types de tours, d'après mon expérience: les défensives et les décoratives

the only information that could resolve the ambiguity of *tours* would come from the incomplete noun phrases on the right of the colon. The recognition of the relation between *tours* and these noun groups would require, among other things, recognizing the complement *d'après mon expérience*, a task that goes beyond the scope of ambiguity reduction and belongs to syntactic parsing. One can say in this case that the context required to disambiguate *tours* is not local. (However, a local context is usually not restricted to a word on each side.)

Fourth, the notions of recall (ability to keep valid analyses) and precision (ability to remove invalid analyses) must be distinguished. Reducing ambiguity requires two aptitudes: recall and precision. Our objective, as defined above, is to increase precision as much as possible. We know that precision cannot always reach 100%, and that it normally falls to parsing to resolve all remaining ambiguity. Let us consider the consequences of rejecting a valid analysis. The output of ambiguity reduction is processed by another component, that can be a parser. We already noticed that the reliability of this component depends on the reliability of the disambiguator: the unavoidable consequence of discarding a valid analysis is the failure of the parsing of a whole sentence. Now, a parser is a program designed to be used as an essential component of translation systems, of speech synthesis from written text, or of

other applications in which reliability of output is an important parameter. Generating reliable output at an acceptable speed is the main purpose of a parser, and the filtering step is introduced only to speed up the process. Maintaining recall at the level of 100% during the step of ambiguity reduction is therefore the highest priority. Maintaining precision at the maximal possible level is the second priority. Because of this order of priority, a lot of caution is advisable in the use of approximations in ambiguity reduction.

5. Data required for ambiguity reduction

Automatic ambiguity reduction involves analysing and recognizing the grammatical context, in order to check local constraints, which are distributional, grammatical, combinatorial constraints on sequences of words or of tags. For instance, the study of sentence (15) exemplifies how we can take advantage of a constraint on noun-adjective agreement. Such constraints, duely encoded, constitute the linguistic data of the system.

Much controversy surrounds the approaches to the construction or acquisition of these data. They are either elaborated by linguists, or obtained by machine learning.

The former approach is chronologically the first (A. Joshi & Ph. Hopely 1996; Z. Harris 1962; Sh. Klein & R. Simmons 1963; B. Greene & G. Rubin 1971). Formally describing grammatical constraints can remain a mere craft or get more or less industrialized, but it requires a non-trivial analytical reflection. Consider the example of *pêcher*, which is ambiguous between a noun and a verb:

(16) *Il a l'impression de ne pêcher que des poissons-chats*

In this sentence, a very local clue indicates that it is a verb: the presence of *ne*. This observation can be stated as:

(17) When it immediately follows *ne*, *pêcher* is a verb

This grammatical constraint correctly discards the nominal tag for *pêcher* from (16). The difficulty lies in the determination of the adequate level of generality. A little general constraint, like (17), applies rarely and resolves few instances of ambiguity. We can generalize it, a natural and intuitive operation, to (18):

(18) Any word that immediately follows *ne* is a verb

This second version still correctly applies to (16), but incorrectly rules out the tag <*plus,ADV*> for *plus* in:

Il a l'impression de ne plus pêcher que des poissons-chats

An excessively general constraint applies to inadequate cases and can reject valid analyses. This undermines the reliability of the disambiguator and of the syntactic parser, if any, and diverts the system from its first-priority objective of maintaining recall at the level of 100%. In order to determine the acceptable level of generality, writers of formal descriptions of constraints have two methods at their disposal:
- searching texts for examples and counter-examples[5], and
- directly constructing examples and especially counter-examples.

[5] Testing the system is a way of automating this search.

These two methods are complementary. The former, though partially automatable, cannot replace the latter. Consider the following example, borrowed from J. Senellart (1999):

(19) In a sequence of the form *(a, as, avions)* <*ADV*> <*V:K*>, the first word is a form of the verb *avoir*

<*V:K*> stands for past participle. This constraint is designed to resolve the ambiguity of *a*, *as*, *avions*, three forms of the verb *avoir* which are ambiguous with nouns. It correctly rules out the noun tag from:

Nous avions également popularisé une technologie

and not a single wrong application of (19) was detected in one year of the newspaper *Le Monde*. Even so, obvious and natural counter-examples exist:

On ne pilote que des avions complètement révisés

in which (19) erroneously removes the noun tag.

The example of (17)-(18) is particularly simple, because it resorts to a very local context. In practice, it is frequently necessary, like in (19), to consider a slightly more extended context, on the left, on the right, or on both sides. The author of the description and formalization of the constraints has to imagine all possible contexts of use of a given word. We insist on the fact that, by definition, parsing tools like systematic recognition of noun phrases and other constituents are not available at that stage.

The second approach to acquiring the linguistic data of disambiguators is generally accepted as the standard technology by the computational-linguistic community. It is based on automatic generalization, or machine learning, from tagged or untagged texts (I. Marshall 1983; J. Benello et al. 1989; B. Merialdo 1994), and, sometimes, from other data like pre-defined rule schemes (E. Brill 1992). This solution is intrinsically approximate. It is oriented towards the processing of cases occurring the most frequently in texts.

The ouptut of automatic generalization can take the form of readable rules, or of unreadable numerical data. In the latter case, the behaviour of the disambiguator is undefined, i.e. the output of the system for a given input can be known by testing the system, but cannot be predicted. In other words, nothing is ensured about the result, and in particular the highest-priority objective of reliably retaining all valid analyses is out of reach.

In addition, very informative tags and contexts encompassing more than one word are difficult to exploit through frequency-based methods. Automatic generalization can indeed be viewed as an exploration of an abstract space the volume of which depends, among other things, on the number of existing tags, on the extent of context taken into account, and on the size of the sample of texts to be explored. An increase in the first parameter may imply an increase in both others. Up to now, the feasibility of testing the approach with fine-grained tags has been limited by the computational complexity involved to explore this space. We mentioned in section 3, for instance, that the separation of senses for adjectives is difficult to deal with in such a framework.

From now on, we will focus on ambiguity reduction data obtained through direct formal description by linguists, and we will consider that a disambiguator is a set of programs but also of linguistic data, including an ambiguity reduction grammar.

6. Examples of grammatical constraints

Formal distributional or grammatical constraints are the main substance of a disambiguator. We will examine examples of issues faced during the elaboration of such data. Each grammatical constraint separately described is sometimes called a rule, and this term cannot always be avoided, though it generally refers to the framework of a network of rules and exceptions. When we use it, however, we will not assume the existence of a system of rule/exception relations. Problems of maintenance are typical of these systems (cf. section 9.1.).

Considering that reliability is an essential quality of a disambiguator, since its output goes on to be exploited by other systems, the definition of our objectives gives priority to ensuring that the process always preserves all valid analyses. This aim is very difficult to achieve. Consider the ambiguity of the words *dément*, verb or adjective; and *visite*, noun or verb:

(20) *Elle dément qu'il s'agisse d'une visite à l'université*

The grammatical words *elle* and *une* are indications that *dément* is a verb and *visite* a noun. We can state the following restriction on the use of these words:

(21) *il, ils, elle, elles* do not occur immediately before an adjective
(22) *un, une* do not occur immediately before a noun

These rules correctly reject the tags[6] <*dément*,*A:ms*> and <*visiter*,*V:P3s*> from (20), but they have counter-examples:

Elle rend celui qui travaille avec elle dément ou génial
Ceux qui voulaient en avoir un affluent à l'entrée du parc

In these sentences, (21) and (22) wrongly rule out <*dément*,*A:ms*> and <*affluer*,*V:P3p*>. A counter-example suffices to disqualify a constraint that discards valid analyses. The difficulty in avoiding this type of mistake is inherent in the problem. On the one hand, writers of grammatical constraints must know all the observable grammatical contexts of the forms to be processed, and take them into account, which (21) and (22) do not. On the other hand, constraints must be stated so as to recognize a sufficient local context in the sentences, without any previous recognition of phrase boundaries, and although this context may be ambiguous. For instance, constraint (21) can be improved by taking into consideration a wider left context — which makes it less general. If an explicit sentence boundary or a subordinating conjunction occurs immediately before <*il*,*PRO*>, (21) applies better. The underlying syntactic fact is the presence of a clause boundary, but grammatical constraints cannot refer to clause boundaries, since analyses are filtered before the syntactic parsing that will, among other things, recognize those boundaries.

Describing grammatical constraints for ambiguity reduction has another unpleasant aspect: it is impossible to complete the description. This was to be expected, since the objective of a complete syntactic description goes beyond the scope of ambiguity reduction. However, it may be impossible to process two very similar cases with the same constraint. Consider again, for instance, constraint (21), that correctly applies immediately after an explicit sentence boundary, like in (20). Syntactically, an adjacency between a pronoun and a verb is a contingent detail of a structure, since a complement may be inserted between *elle* and *dément* without changing the basic sentence:

[6] In conformity with usual Intex notations, the forms included in tags between angle brackets < > are canonical forms.

Elle, sans hésiter, habituée à assumer ses responsabilités, dément qu'il s'agisse d'une visite à l'université

However, constraint (21) does not apply any more, and cannot be adapted so that it applies, since this would mean recognizing completely the complement inserted.

Due to these intrinsic difficulties, formalizing and encoding grammatical constraints is a hard, sometimes frustrating task. One could even claim that the task is unfeasible, arguing that any ambiguity reduction constraint will necessarily have counter-examples; or that the (few) existing systems have already faced these difficulties in all the possible ways, and reached the best possible results. However, these opinions are not based on verifiable facts. On the contrary, we think that the intrinsic difficulties of the problem are well-known, but that solutions have not been investigated systematically. Such an investigation could follow two complementary approaches: on the one hand, elaborating linguistic analyses at the root of the process; on the other hand, building a formalism for stating and applying grammatical constraints. We will examine these two topics successively.

7. Underlying linguistic analyses

Ambiguity reduction always takes place as a step in a global process aiming at assigning a linguistic analysis to sentences of written texts, or several analyses in the case of ambiguous sentences. This goal is in turn a prerequisite for certain types of procedures on written texts. Linguistic analyses are formal descriptions ranging from representations of minimal elements of the text, to syntactic structures of sentences. The question of deciding which analyses are to be assigned to sentences is obviously a fundamental one, though it is little debated in literature. As a matter of fact, the computer data and software components that participate in the process must refer to the same underlying linguistic analyses, which creates an interdependency between these components.

The assignement of a formal structural description to a sentence implies the following steps:

- the elementary units are identified in an electronic dictionary and the corresponding tags are assigned to words, including compound words; at this stage, what we call an analysis or reading of a sentence is a sequence of tags;
- these readings are filtered in order to reduce ambiguity quickly;
- when the selected readings are parsed, relevant constituents and transformations are recognized.

The three steps of the process are dictionary lookup, ambiguity reduction and syntactic parsing. The three modules that implement them rely on a set of linguistic data, respectively an electronic dictionary, an ambiguity reduction grammar, and a syntactic description of the language. Due to conceptual interrelations between these data sets, they must be based on the same analyses. It may happen that several analyses of a given sentence are conceivable: problems can arise when different analyses are chosen for the elaboration of the linguistic data, or when the chosen analysis is formalized in different ways.

Consider for example the following sentence:

Les supporters sont souvent décidés, certains sont même violents

The subject of *sont même violents* can be analysed in two ways. The first solution is to consider that *certains* is a pronoun in this sentence; the electronic dictionary, therefore, must describe it as ambiguous between a determiner, <*certains,DET:mp*> and a pronoun,

<*certains,PRO:mp*>, in addition to one or several adjectives <*certain,A:mp*>. The determiner entry is meant for sentences like:

(23) *Certains supporters sont même violents*

and the pronoun entry for:

(24) *Certains sont même violents*

The other solution consists in representing both cases with the same tag, and describing the syntactic relation between (23) and (24) as a deletion of the noun (M. Gross 1977, pp.28-30). Some representation of this relation of deletion, by the way, is indispensable to a thorough parsing, for other reasons, namely the restitution of the complete subject of (24), no matter which tag is assigned to the word. The second solution is applicable to constructions like *Certains de ces supporters sont même violents* too. If it is adopted, the dictionary represents *certains* as ambiguous between a determiner <*certains,DET:mp*> and one or several adjectives <*certain,A:mp*>. Both solutions are applicable to other determiners: *aucun, deux, trois*... The two solutions are nearly equivalent: the difference between them does not appear as a real divergence in linguistic analysis, but as a formal detail.

However, linguistic concepts cannot be handled by a computer system without formalization, and once underlying linguistic analyses are formalized, a mere formal detail suffices to make them distinct, even if they are equivalent variants. Since the representation of the words at stake in the dictionary depends on the solution chosen, both the ambiguity reduction grammars and the parser must comply with it. Consider now the following grammatical constraint:

(25) A verb cannot occur immediately after a determiner

Constraint (25) may be correct if *certains* is analysed as a pronoun in (24), and indeed it is in keeping with the first solution, but it is incorrect if *certains* is analysed as determiner. In the first solution, *certains* is represented as more ambiguous than in the second. If the second solution is preferred, the ambiguity between <*certains,DET:mp*> and <*certains,PRO:mp*> disappears, as well as the necessity of (25); therefore, the second solution simplifies both the dictionary and the ambiguity reduction grammar.

It is interesting to observe the symptoms that can occur in the case of a discrepancy between the dictionary and the ambiguity reduction grammar.

If the dictionary conforms to the first solution and the grammar to the second, the ambiguity between <*DET*> and <*PRO*> arises and is not resolved by the grammar. This is a clear case of artificial ambiguity, since it is an artifact caused by an inconsistency between two components of the linguistic data.

If, on the contrary, the dictionary conforms to the second solution and the grammar to the first, rule (25) applies and discards the reading in which *certains* is a determiner, keeping only the wrong readings in which *certains* is an adjective. Thus, the right reading is rejected.

The origin of the problem lies in the presence of two formal representations of the same grammatical construct. Another prototypical example of the same situation is the possibility of representing the syntax of a construction either as free or as frozen. The syntax of a construction is said to be free if it is to be treated as a combination of linguistic units, like *good wine*, and frozen if it makes up a compound word, as *white wine*. A form may be ambiguous between a free construction and a compound word, like *table ronde* in (4)-(5). The ambiguity is of a lexical nature, since the tags to be assigned to the form are completely

different depending on the interpretation[7]. This kind of ambiguity can in some cases be resolved locally, which is probably the case of *en fait* in:

En fait, il pleut

where ambiguity is resolved in favour of the compound. Here we are interested in the problems that arise when one can hesitate between two representations, a free one and a frozen one, for the same construction. The syntax of *majuscule* (J. Senellart, 1999) is an example of this situation[8]. All occurrences of *majuscule* correspond to the following forms:
- sequences of the type *N majuscule*, where *N* can only be one of the nouns *lettre, alphabet, caractère, initiale*, or the name of a letter of the alphabet, *a, b*, etc.:

Un seul (*caractère* + * *nom de personne*) *majuscule figure dans le texte*

These forms cannot be related with sentences of the type *N est majuscule*:

* *Ce caractère est majuscule*

The noun *lettre* can always occur in a paraphrase of these forms: *lettre initiale majuscule, alphabet de lettres minuscules*, etc.
- the noun *majuscule* is always equivalent to *lettre majuscule*;
- the form *N être en majuscules* is paraphrasable by *N être écrit en lettres majuscules*.

The syntax of these expressions is so limited that it is best represented by a local grammar (M. Gross 1997). The set of possible values of the noun *N* in the forms *N majuscule* is small: all these forms make up a small family of compound nouns, in which the noun *majuscule* is a reduced form of the compound *lettre majuscule*. Once tags are devised to represent all these compound words, an additional adjectival tag is no longer required for *majuscule*.

A second solution, closer to grammatical and lexicographical traditions, would be to consider that the noun *majuscule* is a simple noun, and the forms *N majuscule* free combinations of nouns with an independent adjective *<majuscule,A>*. The limited syntax of the combination should therefore be represented in the data of the parser, namely the selectional restrictions on *N*, the impossibility of using the "adjective" in a predicative or attributive form, and the possibility of deleting the noun *lettre*.

The difference between the two solutions lies in how the data about the semi-frozen syntax of a family of expressions are dealt with in dictionaries and grammars. These data have to be formally described in any case, but they can appear either in the dictionary (local grammars are a part of the dictionary), or in the grammar of the syntax of the language. In the first solution, the form *majuscule* is not represented as ambiguous, since the tag depends on the noun on the left. In the second solution, the form is systematically considered ambiguous between noun and adjective, and this ambiguity has to be resolved — which complicates the process without any easily identifiable benefit.

The consequences of a discrepancy between the models underlying the dictionary and the ambiguity reduction grammars depend on the type of inconsistency involved. If the dictionary represents the forms as frozen, and the ambiguity reduction grammar considers them free, resolving the ambiguity between noun and adjective in favour of the noun has no undesirable effect, since the word is not tagged as an adjective during lexical analysis. If, on

[7] The chronological order of operations is not different for compound words and simple words: syntactic parsing presupposes that the lexical properties of compound words are previously available, which in turn presupposes that compound words are recognized at the level of lexical analysis.
[8] The example of *terre glaise*, analysed by M. Gross (this volume, section 5.3, example 1), is equivalent.

the contrary, the dictionary represents the forms as free and the grammar considers them frozen, the ambiguity between noun and adjective introduced by lexical analysis is not resolved: the disagreement between the two sets of linguistic data, therefore, gives rise to artificial ambiguity.

We can mention a third example: the gender of adjectives in Romance languages. Some adjectives vary in gender, like *lent/lente*, others do not, like *fixe*. The distinction between masculine and feminine, therefore, appears as relevant grammatical information in the case of *lent*, and as a source of artificial lexical ambiguity in the case of *fixe*, since it leads to handling two tags <*fixe,A:ms*> and <*fixe,A:fs*>. The same holds for various determiners, pronouns and nouns.

The inflectional system of Romance languages, in this respect, can be represented in two superficially different ways. The traditional solution considers that gender is a relevant category for all adjectives, and represents *fixe* as inflectionally ambiguous[9]. In the second solution, gender is relevant for some adjectives and irrelevant for others[10]. The choice between the two solutions depends on the three data sets involved: the dictionary, that represents *fixe* with one adjectival tag or two; the ambiguity reduction grammar, which has to resolve the inflectional ambiguity of *fixe* or not; and the grammar that formalizes the syntax of the language[11]. If *fixe* is represented as ambiguous, the formal representation of all adjectives is more homogeneous, which simplifies the description of noun-adjective agreement. Choosing a solution implies considering the situation as a whole. Needless to say, the coexistence of both models in the same process is bound to cause a major disaster, though they differ only in a mere formal detail.

The three examples: relation between determiners and pronouns, frozen syntax of *majuscule*, gender of adjectives, provide evidence of interdependencies between elements of linguistic description. The construction of electronic dictionaries, of ambiguity reduction grammars, and even the construction of a tagset, require that the same linguistic analyses underly the whole process, including parsing. Problems related to possible inconsistencies or incompatibilities between elements of linguistic description do not invalidate the general architecture that we adopted, namely an organization of data and computer processing into distinct, compatible components. Such practical difficulties are well-known to researchers with an actual descriptive activity in the domain. They only reflect the complexity of real-size linguistic description. The visible outcome of a discrepancy between elements of formal linguistic description can be a decrease in recall or in precision, or the appearance of artificial ambiguity.

The influence of the linguistic analyses underlying formal description is obviously fundamental. Let us point out two stakes depending on the choice of these analyses:

- the simplicity of the global process; choosing globally simple descriptive solutions is an important objective, due to the intrinsic complexity of languages when lexicon is taken into account;
- the degree of lexical ambiguity represented, as measured by the number of tags per word; this number depends on the tags assignable to words in accordance with the linguistic analyses chosen.

In particular, as far as grammatical words are concerned, these two aspects: the simplicity of formal description and the degree of ambiguity of grammatical words, are connected. In the

[9] Combining two tags into <*fixe,A:ms:fs*> or <*fixe,A:mfs*>, for instance, is a notational variant of this first solution.

[10] Authors refer to this solution by using the terms *underspecification* or *portmanteau tags*.

[11] In addition, the choice brings about a difference in the measure of lexical ambiguity, and therefore in the measure of performance of the disambiguator. This example shows how much performance measurement depends on minor formal details (cf. section 3).

example of the relation between determiner and pronoun by noun deletion (23)-(24), the simpler solution is characterized by a smaller number of tags, therefore by a lesser ambiguity, since the problem is treated as syntactic. This example is typical of a frequent situation with grammatical words. When coordination between sets of linguistic data is maximal, the apparent degree of ambiguity decreases since linguistic elements are represented only once.

However, this effect does not suffice to completely remove lexical ambiguity from all grammatical words, nor to make tag assignment useless for grammatical words. Some grammatical words show ambiguity without any synchronic connection with syntactic relations of the type of (23)-(24). For example, *s'* may be a conjunction or pronoun:

(26) *Le public entre s'il n'y a pas de panneaux d'information*
Le vernis s'est endommagé sous l'effet des intempéries

Assigning at least two distinct tags *<si,CONJ>* and *<se,PRO:3>* to the form *s'* has a practical utility: establishing a communication between distinct grammatical constraints. For instance, in (26), a grammatical constraint can safely rule out *<se,PRO:3>* on the grounds of the presence of *il*, and another constraint can use the other tag, *<si,CONJ>*, as a mark of clause boundary. Stating the two grammatical constraints separately is convenient, since there is no connection between them, and only the tag for *s'* makes it possible for the second to use the result of the first. Tags for grammatical words, thus, are a convenient instrument for efficient ambiguity reduction. In addition, several grammatical words are ambiguous with lexical words: the pronoun and determiner *certains* is homograph with plural forms of an adjective, the conjunction *or* with a noun, the preposition *entre* with forms of the verb *entrer*...

Constructing ambiguity reduction grammars is an important activity because it contributes to the laborious, progressive elaboration of formal grammars of languages and testing solutions on texts. In the long term, this could even become the most interesting aspect of ambiguity reduction grammars.

8. Necessity of formalisms for ambiguity reduction

Ambiguity reduction involves:
- describing grammatical constraints during the construction of the system,
- automatically selecting or filtering the readings in conformity with the constraints, when the system is applied to texts.

The description of constraints and the selection of readings cannot be implemented without a description formalism that defines conventions for formalizing and interpreting grammatical constraints.

Linguistic description in general can be more or less formal. Lack of formalism can make description inaccurate, incomplete, incoherent or inapplicable. A high level of formalism, however, does not ensure by itself the adequacy of content: on the contrary, if it means a proliferation of formal details, it becomes an obstacle to description. The issue of formalization or encoding, therefore, conditions the success of the enterprise.

The issue of automating the selection of readings, though it is more computer-related, is not independent of the interpretation of the formal description, since the criteria of selection are the grammatical constraints. One of the functions of a formalism for ambiguity reduction is to provide a basis for implementing a selection module checking the constraints described.

We will discuss and exemplify the reasons why a formalism, or formalisms, are required for reducing ambiguity.

In the absence of a specific formalism, it is impossible to define what would constitute a correct grammatical constraint, since the interpretation of constraints will remain vague. Consider the following grammatical constraint:

(27) Immediately after an <*N:ms*>, an <*A*> in the singular is in the masculine

(27) may be correct or incorrect... because natural languages are so ambiguous. If it is interpreted in the sense that it removes all analyses in which a tag <*N:ms*> occurs immediately before a tag <*A:fs*>, it can have some relevancy; it correctly resolves, for example, gender ambiguity in the following sentence:

On voit apparaître un fichier supplémentaire

If the interpretation is that (27) applies after any word that can be an <*N:ms*>, the rule is wrong, since it erroneously discards the analysis of *sincère* in the feminine in the following sentence:

Voilà une déclaration que l'opinion en général juge sincère

because of the ambiguity of *juge* between verb and noun. The problem here stems from the fact that the grammatical constraint is not specified accurately. Since the exact nature of the grammatical constraint is unclear, it is impossible to tell whether it is valid.

The main function of an ambiguity reduction formalism, then, is very simple: ensuring that the interpretation of constraints described is clear and precise, so that it is possible to define what a correct constraint is (i.e. a constraint which is in accordance with the linguistic analyses underlying the project), and that the operation of filtering according to correct constraints can be automated.

It is logical and legitimate to try to design the simplest possible formalism. For instance, a minimal formalism, almost an absence of formalism, would consist in stating grammatical constraints in the form of mere sets of grammatical sequences, exactly as if they were local grammars describing families of forms. Adjective-noun agreement could be written as:

(28) <*N:ms*> <*A:ms*> + <*N:mp*> <*A:mp*> + <*N:fs*> <*A:fs*> + <*N:fp*> <*A:fp*>

and compound tenses of verbs with auxiliary *avoir* would appear as:

<*avoir,V*> <*V:K*>

etc. Unfortunately, the idea is too simple. Descriptions of forms do not explicitly define grammatical constraints because they do not explicitly rule out any analyses. In this respect, (28) is even less explicit than (27). In order to interpret a description of forms like (28) as a grammatical constraint unambiguously, we would need conventions for its interpretation. Several conventions would be possible; they would lead to distinct interpretations, i.e., to distinct grammatical constraints. Different interpretation conventions would have advantages and drawbacks for description writers; in order to be used, they would have to be implemented by different computer programs. They would therefore be distinct formalisms, unavoidably more complex than the initial idea.

The preceding example stresses the fact that explicitly specifying the effects of the application of a grammar to a text is the responsibility of a formalism for ambiguity reduction.

As a matter of fact, a formalism cannot be used satisfactorily without such a specification, which predicts which type of analyses are actually rejected when one applies grammatical constraints stated with the formalism. A specification serves as a common reference point for grammar writers, which describe constraints, and computer program writers that implement and maintain the system. The usefulness of a specification becomes obvious in the case of an error in one of these two parts of the system, either in linguistic data, or in the program. When an error is discovered, the specification is used in order to tell which part it stems from: for instance, it may happen that the procedure of filtering performed by the program disagrees with the specification. As long as the error is not corrected, the specification is maintained as a common reference point, and the other part is not disturbed. In the absence of a specification, both parts of the system are disturbed: the system does not behave in conformity with some implicit reference point, but its future behaviour is unknown until the error is corrected. Software engineers systematically resort to specifications for the construction of complex computer systems: they are a way of organizing communication between teams and between complex parts of a system. Obviously, a specification does not resolve all problems: it can change, which disturbs the whole system, but this happens less often than programming mistakes or errors in linguistic description. Thus, an ambiguity reduction formalism must ensure that grammatical constraints stated with it are accurately and unambiguously defined.

The main quality of a formalism is probably its simplicity, but we saw that the problem involved is not so simple that one could manage without any formalism at all. The comfort granted by disambiguation formalisms to writers of grammatical constraints is the main approach to defining criteria of quality:

- simplicity and readability,
- predictability of the effect of applying a grammar to a text,
- possibility of organizing a grammar into small components relatively independent of one another,
- possibility of automating the specified operations efficiently.

We will mention three other features that, according to our experimentations in constructing and using ambiguity reduction grammars (É. Laporte & A. Monceaux 1999), are relevant to the quality of formalisms.

- Punctuation tags and variable tags, i.e. tags that represent categories of words, like parts of speech, e.g. <*N*>, are indispensable for encoding the description of the context of many constraints. All examples of constraints stated above, for instance, include tags of this type, except constraint (17).
- A formalism may have more or less expressive power, in the sense that it provides more or less technical means of stating grammatical constraints of such or such kind. The notion of expressive power is exemplified in section 9.
- In the case of overlap between the application zones of one or several grammatical constraints in a text[12], the effect of applying the constraints depends on the specification associated with the formalism. Two modes of operation are imaginable and have already been tested in actual systems: the application of a constraint to a given zone may depend, or not, on the possible presence of an overlap with another application zone. The first mode is that of systems that, in the case of overlap between two application zones, only check the constraint in one of the two zones, e.g. the left one. With this convention, two grammatical constraints can yield a correct result when they are applied separately, but be incompatible when two application zones happen to overlap. It is even theoretically possible that, when you add a constraint to an existing ambiguity reduction grammar, the rate of ambiguity reduction

[12] In sentence (26), for example, we mentioned the possibility for a grammatical constraint to resolve the ambiguity of *s'*, and for another to use the same *s'*, tagged as a conjunction, as a mark of clause boundary. In such a situation, the form *s'* would belong in the respective application zones of both constraints.

decreases, since it blocks some instances of application of constraints already present in the grammar. With this mode of operation, the result of applying a grammatical constraint does not depend only on the context explicitly described in it, but also on the contexts of other constraints that can have some overlap with it. The presence of overlaps extends the context that determines the result of application, and de facto changes the interpretation of the grammatical constraint in function of interactions between neighbouring application zones.

In the second mode of operation, the formalism itself ensures that the result of applying each constraint to each application zone is independent of the rest of the text, on the left as well as on the right. This does not preclude the apparent "co-operation" between rules exemplified by sentence (26): the grammatical constraint that uses *<si,CONJ>* as a mark of clause boundary does not apply to analyses with *<se,PRO:3>*, and, therefore, does not remove them; they are independently removed by the other constraint, due to the presence of the verb on the left.

9. A typology of ambiguity reduction formalisms

We have mentioned that an ambiguity reduction formalism simultaneously defines conventions for the description of grammatical constraints and for the result of the application of the constraints to analyses generated by dictionary lookup. These two aspects are closely connected.

Approaches to filtering, for example, can be positive or negative, depending whether you describe analyses to be kept or analyses to be discarded. This option has consequences not only on the description of constraints but also on the specification of the filtering. Both options are theoretically possible, and have been implemented in actual systems. From the point of view of application, they do not provide the same advantages.

In the light of the several experimentations carried out at IGM in the recent years, we will present a typology of ambiguity reduction formalisms and comment upon their respective potential.

We will classify ambiguity reduction formalisms into two types on the basis of the following criterion of distinction. We will consider the elementary automatic operation by which a given analysis of a given sentence is rejected or preserved as an application of a given rule described in the formalism. Such an elementary decision depends on the conditions encoded in the rule. Conceptually, the whole process of filtering can be viewed as a repetition of instances of this elementary decision mechanism. Depending on the formalism used, that operation can take into account conditions about:
- the whole set of alternative analyses of the sentence, including the analysis at stake, or
- only the analysis at stake, independently of all others.

We will call formalisms of the first type dependent-filtering formalisms, and others independent-filtering ones. Examples of formalisms of both types are described in literature. We apologize for introducing such strange terms as dependent and independent filtering, but no term has been proposed for the first type yet, and the term already proposed for the second is *monotonous* (K. Koskenniemi 1990), which is in fact more evocative of the general approach which consists in filtering analyses by applying formal constraints.

9.1. Dependent-filtering formalisms

Our definition of this type of ambiguity reduction formalisms is that the decision to reject or keep a given analysis of a sentence can depend on that analysis, as well as on other analyses of the same sentence, or at least on those that do not have been eliminated by previously applying other rules. An example of such a behaviour is analysed in É. Laporte & A.

Monceaux (1999, pp. 357-359). When a grammatical constraint recognizes a grammatical sequence in a text (what we will call a target), it can rule out analyses belonging to the target as well as alternative analyses of the same word sequence. Several disambiguators are based on dependent-filtering formalisms (M. Silberztein 1994; A. Voutilainen 1994; É. Tzoukermann et al. 1995); in Ph. Laval (1995), one of the two formalisms described (p. 103) is of the same type.

In order to assess the potential of this type of formalism for applications, let us examine some formal properties deducible from its definition.

The first property is the existence of interactions and interdependencies among constraints included in the same grammar. Since the decision to remove or retain a given analysis of a given sentence may depend on other analyses of the same sentence that would not have been rejected by applying other rules before, the result of applying a grammatical constraint depends on other constraints previously applied. In other words, the interpretation and effects of a grammatical constraint may be different in the absence or in the presence of another grammatical constraint.

This first property has three consequences.

a) When you add new constraints to an existing ambiguity reduction grammar that has a known performance, the rate of reduction may decrease instead of increasing, since if a new constraint applies to the text and blocks some constraint of the former version, the new constraint may discard less analyses than the former.

b) The objective of maintaining recall at the level of 100% gets more and more difficult to reach as grammatical constraints accumulate, since when you introduce a new constraint into an existing grammar, the effects of others can change, so that constraints that used to keep valid analyses can now rule them out.

c) It is possible to determine whether the application of a complete grammar to a given input is correct or not: you just compare the output with the linguistic analysis you decided to adopt. But, in the framework of a dependent-filtering formalism, the application of two correct grammars is not certain to produce correct output, since the result of applying a given constraint to sentences depends on the other constraints applied before. Therefore, even if the effects of applying constraints are explicitly specified with the formalism, it is not possible to define what a correct grammatical constraint is.

The second property of dependent-filtering formalisms is the necessity to define one or several ways of combining grammatical constraints. Indeed, we already mentioned that the effect of applying a constraint to a sentence has to be defined; now, it is equally necessary to define the effect of applying several constraints formalized separately. Two kinds of modes of combination are possible:

a) With combination by composition, the output of a constraint is the input to the next constraint. This mode of combination involves a composition of functions in the mathematical sense. Final output depends on the order of composition, as is easy to check with a priority rule, for instance (É. Laporte & A. Monceaux 1999, p. 357). A variant of this mode of combination consists in iterating the application of a series of rules, always in the same order, until a complete series of applications has no effect: then the set of analyses obtained is called a fixed point, because it cannot be changed any more by applying the same series of rules again. There is a mathematical certainty that a fixed point is reached after a finite number of iterations, because all sets of analyses of a given sentence are finite and can only be reduced. However, it is easy to check that the fixed point obtained depends on the order of composition, i.e. the process can reach several different fixed points from the same text, depending on the order of composition.

b) Modes of simultaneous combination can also be defined. The simplest is the following: the result of applying a set of rules to a sentence is defined as the set of analyses

that would be kept by all rules if each of them was applied to the same set of analyses as if none of the other rules existed. When all constraints are applied in this way, all interactions between rules disappear; defining what a correct grammatical constraint is becomes possible, and the formalism ensures that an accumulation of correct constraints is correct; the construction of an ambiguity reduction grammar by accumulation of grammatical constraints becomes simpler and safer; and the application of constraints is not restricted by conditions regarding the disposition of application zones of other constraints. In fact, the formal properties of this kind of formalism make it close to independent-filtering formalisms.

c) Other modes of simultaneous combination can be defined on the basis of conventions determining whether an analysis discarded by a constraint and preserved by another is discarded or preserved, according to various criteria, e.g. the disposition of application zones and overlaps between them, or a network of rule/exception relations between grammatical constraints. Such modes of simultaneous combination introduce new interactions and dependencies between grammatical constraints, with the consequences mentioned above. In the example of a network of rules and exceptions, the decision about a given analysis depends on conditions regarding other rules applied to the same analysis and marked as exceptions. Dependency relations among grammatical constraints make it difficult to maintain and extend the grammar, since any modification or new element can modify the behaviour of all constraints linked by direct or indirect dependency relations.

Whatever the mode of combination of rules, a detailed and complex specification is required. When such a specification exists, it is possible to define what a correct system of constraints is, but even so it is not always possible to define what a correct grammatical constraint is. In addition, writers of ambiguity reduction grammars must understand and digest the specification, since the correctness of their grammars depends on it. This is a cause of errors, because the description of grammatical constraints is based on linguistic intuition: intuitions are consistent with what writers think the system does, and if they did not correctly digest the formalism, their intuitions of grammatical constraints can be wrong. Of course, this is the main pragmatic reason why simplicity is an essential quality of a formalism.

9.2. Independent-filtering formalisms

Our definition of this type of ambiguity reduction formalisms is that the decision of ruling out or retaining a given analysis of a given sentence depends only on the analysis at stake, not on other analyses of the same sentence. These formalisms, therefore, have less expressive power, since the conditions of application of grammatical constraints can only include properties of the analysis in question.

The following grammatical constraints (E. Roche 1992) belong to this kind of filtering:

(29) <*le,DET*> does not occur before a verb at a finite tense
(30) <*le,PRO*> does not occur before a noun

These constraints correctly remove <*le,DET:fs*> from (31) and <*le,PRO:3fs*> from (32):

(31) *Il la prononce naturellement*
(32) *La prononciation est naturelle*

When a verb and a noun are homographs, (29) and (30) correctly reject the analyses in <*le,DET:fs*> <*V:3s*> and <*le,PRO:3fs*> <*N:fs*>:

On recherche par quel service la fiche a transité

The analyses with *<le,DET:fs> <fiche,N:fs>* and *<le,PRO:3fs> <ficher,V:3s>* are preserved. This is an example of independent filtering because (29) applies just to the analyses with the verb, and (30) just to those with the noun, as though the processing of each analysis were entirely independent. When a constraint recognizes a target, it can only remove analyses belonging to this target, not alternative analyses. In other words, when a constraint is written in order to resolve a type of ambiguity, the writer does not need to care about others. Because of this limitation of the expressive power of the formalism, some options are useless, e.g. recognizing a context only if all ambiguity has already been resolved in it. If a writer of grammatical constraints elaborates the recognition of the target, making it safer and more precise, the output of the rule itself, i.e. the selection of analyses to be discarded, is safer and more precise. Writers can thus achieve a better control on their rules.

In addition to E. Roche 1992, one of the two formalisms proposed by Ph. Laval (1995, p. 101) and ELAG (É. Laporte & A. Monceaux 1999) are examples of independent-filtering formalisms. The system of A. Voutilainen (1994) allows the user to build independent-filtering disambiguators by restricting himself to a part of the expressive power of the formalism, e.g. by writing REMOVE rules and avoiding SELECT rules. All these systems are based on finite automata and explore general approaches defined by M. Gross (1989) and K. Koskenniemi (1990).

The effect of the application of a combination of grammatical constraints is simple to define and does not present the variants we observed in the case of dependent-filtering formalisms: an analysis is kept by the ambiguity reduction grammar if, and only if it is kept by each of the grammatical constraints included in it. The fixed point in the iterative application of rules, for example, is reached after the first application, which makes the notion of fixed point useless.

An ambiguity reduction grammar stated with an independent-filtering formalism has automatically two other important applications.

- It can be used to detect non-lexical errors: when a grammar is correct and rejects all analyses of a sentence, this implies that the sentence does not have any valid analysis, and therefore that it is ill-formed. In the case of a dependent-filtering formalism, this secondary use is hardly likely to work, because invalid analyses are frequently recognized through the existence of alternative analyses, which may be valid or invalid; now, this situation of ambiguity has no reason to exist in the case of non-lexical errors.

- It can be used to resolve homophonies of the type of *freine/frêne*, which disturb speech recognition due to the lack of phonetic reasons for selecting the right spelling. For instance, if a word is transcribed *freine/frêne*, and if an ambiguity reduction grammar establishes by exploring the context that the word is a verb, the hypothesis *frêne* can be ruled out. Independent-filtering formalisms allow for such a use because they are based on a discrimination between right and wrong grammatical analyses, no matter whether forms are ambiguous. The same secondary use with dependent-filtering formalisms is much less realistic, since they organize description on the basis of sets of homographic forms, which are quite different from sets of homophonous forms.

Conclusion

Ambiguity reduction seems to be a simple task at first sight, but the examples above make it obvious that one can realistically expect more consistent, reliable and useful results than those now considered successful. Such progress will not be reached without considering more elaborate linguistic analyses: the informative content of linguistic data is as important here as

it is in other fields of computational linguistics. In addition to this point, we hope we showed that a reflection about the formal and abstract aspects of the problem, too, is absolutely necessary if we want to design and implement so simple and efficient tools that they allow for elaborating good ambiguity reduction grammars.

Author's address

Éric Laporte
Institut Gaspard-Monge
Université de Marne-la-Vallée - CNRS
5, bd Descartes
F-77454 Marne-la-Vallée CEDEX 2
France

REFERENCES

Benello, J.; A.W. Mackie; J.A. Anderson. 1989. Syntactic category disambiguation with neural networks. *Computer Speech and Language* 3, London: Academic Press, pp. 203-217.

Brill, Eric. 1992. A simple rule-based part-of-speech tagger. In *Proceedings of the 3th Conference on applied Natural Language Processing*, Trento, Italy, pp. 152-155,.

Cloeren, Jan. 1999. Tagsets. In *Syntactic Wordclass Tagging*, H. van Halteren (ed.), Dordrecht/Boston/London: Kluwer, pp. 37-54.

Francis, Nelson W.; Henry Kucera. 1982. *Frequency Analysis of English Usage: Lexicon and Grammar*. Boston: Houghton Mifflin.

Garrigues, Mylène. 1997. Une méthode de désambiguïsation locale nom/adjectif pour l'analyse automatique de textes. In *Langages* 126, *La description syntaxique des adjectifs pour les traitements informatiques*, Nam Jee-sun (ed.), Paris: Larousse, pp. 60-78..

Garside, Roger; Geoffrey Leech; Geoffrey Sampson (eds.). 1987. *The computational analysis of English: a corpus-based approach*. Harlow: Longman.

Greene, Barbara B.; Gerald M. Rubin. 1971. *Automated grammatical tagging of English*. Providence, Rhode Island.

Gross, Maurice. 1977. *Grammaire transformationnelle du français. Syntaxe du nom*, Paris: Larousse, 256 p.

Gross, Maurice. 1989. The use of finite automata in the lexical representation of natural language. In *Lecture Notes in Computer Science* 377, *Electronic Dictionaries and Automata in Computational Linguistics*, M. Gross & D. Perrin (eds.), Berlin: Springer, pp. 34-50.

Gross, Maurice. 1994a. Constructing Lexicon-Grammars. In *Computational approaches to the Lexicon*, B.T.S.Atkins, A. Zampolli (eds.), Oxford Univ. Press, pp. 213-163.

Gross, Maurice. 1994b. "Lexicon-Grammar: Application to French". In *The Encyclopedia of Language and Linguistics*, R.E.Asher (ed.), Oxford/NewYork/Seoul/Tokyo: Pergamon, vol. 4, pp. 2195-2205.

Gross, Maurice. 1997. The construction of local grammars. In *Finite-State Language Processing*, E. Roche & Y. Schabès (eds.), Cambridge, Mass.: MIT, pp. 329-352.

Harris, Zellig S. 1962. *String Analysis of Language Structure*. Mouton.

Johansson, Stig; Eric S. Atwell; Roger Garside; Geoffrey Leech. 1986. *The Tagged LOB Corpus: Users' manual*. ICAME, The Norwegian Computing Centre for the Humanities, Bergen University, Norway.

Joshi, Aravind K.; Philip Hopely. 1996. A Parser from antiquity, *Natural Language Engineering* 2(4), Cambridge Univ. Press, pp. 291-294.

Klein, Sheldon; Robert F. Simmons. 1963. A Computational approach to grammatical coding of English words. *Journal of the Association for Computing Machinery* 10(3), pp. 334-337.

Koskenniemi, Kimmo. 1990. Finite-state parsing and disambiguation. In *Proceedings of COLING-1990*, H. Karlgren (ed.), Helsinki, pp. 229-232.

Laporte, Éric; Anne Monceaux. 1999. Elimination of lexical ambiguities by grammars: the ELAG system. *Lingvisticae Investigationes* XXII, Cédrick Fairon (ed.), Amsterdam/Philadelphie: Benjamins, pp. 341-367.

Laval, Philippe. 1995. Un système simple de levée des homographies. *Lingvisticae Investigationes* XIX(1), Amsterdam/Philadelphie: Benjamins, pp. 97-105.

Leech, Geoffrey; Roger Garside; Michael Bryant. 1994. CLAWS4: The Tagging of the British National Corpus. In *Proceedings of COLING-94*, Kyoto.

Leech, Geoffrey; Andrew Wilson. 1999. Standards for Tagsets. In *Syntactic Wordclass Tagging*, H. van Halteren (ed.), Dordrecht/Boston/London: Kluwer, pp. 55-80.

Marcus, Mitchell P.; Beatrice Santorini; Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics* 19(2), Cambridge, Mass.: MIT, pp. 315-330.

Marshall, Ian. 1983. Choice of grammatical word-class without global syntactic analysis: Tagging words in the LOB Corpus. *Computers and the Humanities* 17, Dordrecht/Boston/London: Kluwer, pp. 139-150.

Merialdo, Bernard. 1994. Tagging English text with a probabilistic model. *Computational Linguistics* 20(2), Cambridge, Mass.: MIT, pp. 155-171.

Milne, Robert W. 1986. Resolving lexical ambiguity in a deterministic parser. *Computational Linguistics* 12(1), Cambridge, Mass.: MIT, pp. 1-12.

Roche, Emmanuel. 1992. Text disambiguation by finite state automata, an algorithm and experiments on corpora. In *Proceedings of COLING-92*, Nantes.

Salkoff, Morris. 1999. A study of ambiguity using Intex. *Lingvisticae Investigationes* XXII, Cédrick Fairon (ed.), Amsterdam/Philadelphie: Benjamins, pp. 143-154.

Senellart, Jean. 1999. *Outils de reconnaissance d'expressions linguistiques complexes dans de grands corpus*. PhD thesis, LADL, University Paris 7, 290 p.

Silberztein, Max. 1994. INTEX: a corpus processing system. In *Proceedings of COLING-94*, Kyoto.

Tzoukermann, Évelyne; Dragomir R. Radev; William A. Gale. 1995. Combining linguistic knowledge and statistical learning in French part-of-speech tagging. In *Proceedings of the SIGDAT Workshop "From Texts to Tags: Issues in Multilanguage Analysis"*, European Chapter of the ACL, Dublin.

Voutilainen, Atro; Pasi Tapanainen. 1993. Ambiguity resolution in a reductionistic parser. In *Proceedings of the 6th Conference of the European Chapter of the ACL*, Utrecht, pp. 394-403.

Voutilainen, Atro. 1994. Morphological disambiguation. In *Constraint Grammar: a Language-Independent System for Parsing Unrestricted Text*, F. Karlsson, A. Voutilainen, J. Heikkilä, A. Attila (eds.), Berlin/New York: Mouton-de Gruyter.

SUMMARY

We examine various issues faced during the elaboration of lexical disambiguators, e.g. issues related with linguistic analyses underlying disambiguators, and we exemplify these issues with grammatical constraints. We also examine computational problems and show how they

are connected with linguistic problems: the influence of the granularity of tagsets, the definition of realistic and useful objectives, and the construction of the data required for the reduction of ambiguity. We show why a formalism is required for automatic ambiguity reduction, we analyse its function and we present a typology of such formalisms.